What is claimed is:

1. A logic system in a data packet processor for selecting and releasing a context among a plurality of such contexts, the selected and released ccontext dedicated for enabling processing of interrupt service routines corresponding to interrupts generated in data packet processing and pending for service, the system comprising:

a first determination logic for determining control status of all of the contexts;

a second determination logic for determining if a context is idle or not;

a selection logic for selecting a context; and

a context release mechanism for releasing the selected context;

characterized in that determination by the logic system that all contexts are singularly owned by an entity not responsible for packet processing and that at least one of the contexts is idle, triggers immediate selection and release of one of the at least one idle contexts to an entity responsible for packet processing.

2. The logic system of claim 1 wherein the logic system, upon determination that all contexts are owned by the entity not responsible for packet processing and that there are no idle contexts, all contexts being pre-loaded, aborts pre-loading of one of the contexts to render that context idle.

3. The logic system of claim 1 wherein the data packet processor is part of a data packet routing system.

4. The logic system of claim 3 wherein the data packet routing system is connected to a data packet network.

5. The logic system of claim 4 wherein the data packet network is the Internet network.

6. The logic system of claim 1 wherein the first and second determination logics and the selection logic and release mechanism are integrated on one hardware device.

7. The logic system of claim 6 wherein the hardware device is a register transfer unit.

8. The logic system of claim 1 further comprising a memory marker pointing to a memory address of a pre-defined instruction thread for invoking a stream to run in the released context.

9. The logic system of claim 1 wherein the selected and released context is dedicated to the servicing of pending interrupts.

10. The logic system of claim 8 wherein the memory containing the pre-defined thread is a cache memory used for containing program instructions.

11. The logic system of claim 8 wherein the memory containing the pre-defined thread is a dedicated memory section of a processing core, the core also containing the contexts.

12. The logic system of claim 1 wherein a priority scheme is used to select an idle context in the case of more than one idle context.

13. The logic system of claim 1 wherein the priority rules used are based on prediction of likely context assignment of contexts released to the processing entity after release of the context for servicing interrupts.

14. A method for selecting and releasing a context among a plurality of contexts to a processing entity for processing interrupt service routines associated with interrupts generated during data packet processing, the method comprising steps of:

(a) determining that none of the contexts are under control of an entity other that the processing entity;

(b) determining that there is at least one idle context within the singularly controlled group of contexts;

(c) selecting one of the at least one idle contexts for release; and

(d) releasing the selected context to the processing entity, so the processing entity may process service routines.

15. the method of claim 14 further comprising a step (e) for processing low-priority tasks before releasing the selected context.

16. The method of claim 14 wherein the data packet processing is performed by a processor of a data packet router connected to a data packet network.

17. The method of claim 15 wherein the data packet network is the Internet network.

18. The method of claim 14 wherein in step (a) the determination of control status of the contexts is made by logic on a hardware device.

19. The method of claim 14 wherein in step (b) the determination of idle status of the contexts is made by logic on a hardware device.

20. The method of claim 14 wherein in steps (a) and (b) are enabled by a single hardware device containing the appropriate logics.

21. The method of claim 20 wherein in steps (c) and (d) are also enabled by the same hardware device containing the appropriate logics.

22. The method of claim 14 wherein in step (d), a memory marker is sent along with release notification, the marker pointing to a location in memory where a pre-defined instruction thread exists to be used by the processing entity for invoking a stream to run in the released context.

23. A method for processing service routines associated in a data packet processing system using a dedicated context selected among a plurality of contexts not under control of the processor comprising steps of:

(a) receiving, at the processor, notification of a selected context about to be released and memory marker pointing to an instruction thread in memory;

(b) fetching the instruction thread designated by the received marker;

(c) executing the instruction thread in the released context now under processor control;

(d) detecting at least one pending interrupt that requires processing; and

(e) running the corresponding service routine or routines that satisfies the at least one interrupt.

24. The method of claim 23 further comprising a step for calling and executing other service routines after all interrupts are serviced.

25. The method of claim 23 wherein the data processing system is a data packet router connected to a data packet network.

26. The method of claim 25 wherein the data packet network is the Internet network.

27. The method of claim 23 wherein in step (a) the instruction thread has at least one instruction invoking a processing stream.

28. The method of claim 23 wherein in step (a) the memory marker is a program counter number of the beginning location of the thread in memory.

29. The method of claim 23 wherein in step (a) the memory is an instruction cache memory containing a program of instructions.

30. The method of claim 23 wherein in step (a) the memory is a dedicated memory in the processing core, the core also containing the contexts.

31. The method of claim 23 wherein in step (b) fetching the instruction thread occurs before the context is actually under the control of the processor.

32. The method of claim 23 wherein in step (c) the instruction thread contains at least one instruction invoking a stream.

33. The method of claim 32 wherein in step (c) the instruction thread further contains an instruction for checking for pending interrupts.

34. The method of claim 23 wherein in step (d) the interrupts are taken by priority in the event of more than one pending.

35. The method of claim 23 wherein steps (a) through (e) are completed when no other contexts are under control of the processor.